In the Claims:

Please amend Claims 1, 2, 5, 6, 7, 11, 12, 15, 20, 21, 25, 27, 28, 31-33, 37, 38, 41, 43, 46 and 51; cancel Claims 16 and 42; and add new Claims 53 and 54; all as shown below. Applicant reserves the right to prosecute any originally presented or canceled claims in a continuing or future application.

1. (Currently Amended) A framework architecture system for allowing a client application to communicate with a server component application, comprising:

a server having a server engine ~~for providing~~ that provides client access to the server, said server engine further including[[:]]

a server component ~~for providing~~ that provides a service;

an implementation within said server component ~~for providing~~ that provides functions of said service, wherein said implementation is dynamically linked and loaded into the server engine, so that the server engine is not reconfigured and recompiled; and,

an interface ~~mechanism for allowing~~ that allows a client application to access said implementation, wherein said interface is dynamically customized and loaded into the clients address space.

2. (Currently Amended) The framework architecture system of claim 1 wherein said server component includes a plurality of implementations, and wherein said interface ~~mechanism~~ allows said client application to select and access one of said ~~plugin~~ implementations.

3. (Original) The framework architecture system of claim 1 wherein said implementation is a plugin implementation and can be replaced by another implementation at run time.

4. (Original) The framework architecture system of claim 1 wherein said interface provides an interface definition specification for the functions provided by said implementation of said server component.

5. (Currently Amended) The framework architecture system of claim 4 wherein said interface presents a data type structure according to said interface definition specification of which each member of said data type structure is a function pointer to the implementation functions implementing the services defined by said interface.

6. (Currently Amended) The framework architecture system of claim 1 further comprising:

an interface namespace containing a record for ~~each~~ said interface together with an interface identifier.

7. (Currently Amended) The framework architecture system of claim 1 wherein ~~each~~ said interface is associated with a version ~~of said interface~~.

8. (Original) The framework architecture system of claim 1 wherein there exists multiple implementations for said interface.

9. (Original) The framework architecture system of claim 1 wherein there exists multiple interfaces providing the same implementation for said server component.

10. (Original) The framework architecture system of claim 1 wherein said implementation inherits from another implementation of the same interface a subset of interface methods and functions.

11. (Currently Amended) The framework architecture system of claim 3 wherein implementations are plugged into said engine and removed from said server engine by a registration process.

12. (Currently Amended) The framework architecture system of claim 1 wherein said implementation is stored in a container that is loaded by the server engine at run-time

13. (Original) The framework architecture system of claim 12 wherein said container is a dynamic loadable library.

14. (Original) The framework architecture system of claim 12 wherein said container contains multiple plugins.

15. (Currently Amended) The framework architecture system of claim 1 wherein said implementation ~~may include~~ includes an interceptor for adding services to said server component.

16. (Canceled)

17. (Original) The framework architecture system of claim 15 wherein said interceptor is a stack interceptor which during the realization of an interface implementation causes each plugin in an interception sequence to be instantiated in turn.

18. (Original) The framework architecture system of claim 1 further including a registry for persistent storage of implementations.

19. (Original) The framework architecture system of claim 1 further including a realization mechanism for allowing a client application to realize an implementation, wherein said implementation includes a v table, private data store, and per-instance data structure.

20. (Currently Amended) The framework architecture system of claim 19 wherein the realization mechanism includes logic for retrieving information stored within the ~~plugins~~ v table, private data store, and per-instance data structure, copying said information to a proxy v table, and returning a pointer to the proxy v table to the client.

21. (Currently Amended) The framework architecture system of claim 20 wherein the client uses this pointer to thereafter communicate with the ~~interface~~ implementation.

22. (Original) The framework architecture system of claim 1 wherein said implementation is a software personality.

23. (Original) The framework architecture system of claim 22 wherein said software personality is one of Tuxedo, Jolt, or AMS.

24. (Original) The framework architecture system of claim 23 wherein said personality includes a programming attitude.

25. (Currently Amended) The framework architecture system of claim 24 wherein said programming attitude is one [[os]] of C, OLE, or Cobol.

26. (Original) The framework architecture system of claim 1 wherein said implementation is a software extension.

27. (Currently Amended) A method of allowing a client application to communicate with a server application via a framework architecture, comprising the steps of:

providing a server engine ~~for providing~~ that provides client access to ~~the~~ a server, said server engine further including:

a server component ~~for providing~~ that provides a service;

an implementation within said server component ~~for providing~~ that provides functions of said service, wherein said implementation is dynamically linked and loaded into the server engine, so that the server engine is not reconfigured and recompiled; and,

an interface ~~mechanism for allowing~~ that allows a client application to access said implementation, wherein said interface is dynamically customized and loaded into the clients address space.

28. (Currently Amended) The method of claim 27 ~~further including~~ wherein:

said server component includes a plurality of implementations, and wherein said interface ~~allowing~~ allows said client application to select and access one of a plurality of ~~plugin~~ implementations provided by said interface.

29. (Original) The method of claim 27 wherein said implementation is a plugin implementation and can be replaced by another implementation at run time.

30. (Original) The method of claim 27 wherein said interface provides an interface definition specification for the functions provided by said implementation of said server component.

31. (Currently Amended) The method of claim 30 wherein said interface presents a data type structure according to said interface definition specification of which each member of said data type structure is a function pointer to the implementation functions implementing the services defined by said interface.

32. (Currently Amended) The method of claim 27 further comprising:

maintaining an interface namespace containing a record for ~~each~~ said interface together with an interface identifier.

33. (Currently Amended) The method of claim 27 further including:

associating ~~each~~ said interface with a version ~~of said interface~~.

34. (Original) The method of claim 27 wherein there exists multiple implementations for said interface.

35. (Original) The method of claim 27 wherein there exists multiple interfaces providing the same implementation for said server component.

36. (Original) The method of claim 27 wherein said implementation inherits from another implementation of the same interface a subset of interface methods and functions.

37. (Currently Amended) The method of claim 29 further including:

registering an implementation as a plugin in said server engine via a registration process.

38. (Currently Amended) The method of claim 27 wherein said implementation is stored in a container that is loaded by the server engine at run-time.

39. (Original) The method of claim 38 wherein said container is a dynamic loadable library.

40. (Original) The method of claim 38 wherein said container contains multiple plugins.

41. (Currently Amended) The method of claim 27 further including:

combining said implementation ~~function~~ with an interceptor that adds services to said server component.

42. (Canceled)

43. (Currently Amended) The method of claim 41 wherein said interceptor is a stack interceptor which during the realization of an interface implementation includes the step of:

~~causes~~ causing each plugin in an interception sequence to be instantiated in turn.

44. (Original) The method of claim 27 further including storing implementations within a registry.

45. (Original) The method of claim 27 further including:

providing a realization mechanism for allowing a client application to realize an implementation, wherein said implementation includes a v table, private data store, and per-instance data structure.

46. (Currently Amended) The method of claim 45 further including:

retrieving information stored within the ~~plugins~~ v table, private data store, and per-instance data structure, copying said information to a proxy v table, and returning a pointer to the proxy v table to the client.

47. (Original) The method of claim 46 further including:

using said pointer to thereafter communicate with the interface implementation.

48. (Original) The method of claim 27 wherein said implementation is a software personality.

49. (Original) The method of claim 48 wherein said software personality is one of Tuxedo, Jolt, or AMS.

50. (Original) The method of claim 49 wherein said personality includes a programming attitude.

51. (Currently Amended) The method of claim 50 wherein said programming attitude is one [[os]] of C, OLE, or Cobol.

52. (Original) The method of claim 27 wherein said implementation is a software extension.

53. (New) A framework architecture system for allowing a client application to communicate with a server component application, comprising:

a server engine in a server that provides client access to the server, said server engine further including

a server component that provides a service;

an implementation within said server component that provides functions of said service, wherein said implementation includes an interceptor for adding services to said server component, wherein said interceptor is a fanout interceptor which during the instantiation of an interface implementation causes a plurality of intercepting plugins as specified by an interception sequence attribute to be also instantiated, and wherein subsequent method invocations by the client results in invocation of the corresponding methods of intercepting plugins in the order specified; and,

an interface that allows a client application to access said implementation.

54. (New) A method of allowing a client application to communicate with a server application via a framework architecture, comprising the steps of:

providing a server engine that provides client access to a server, said server engine further including

a server component that provides a service;

an implementation within said server component that provides functions of said service, said implementation further including an interceptor that adds services to said server component, wherein said interceptor is a fanout interceptor which during the instantiation of an interface implementation includes the step of causing a plurality of intercepting plugins as specified by an interception sequence attribute to be also instantiated, and wherein subsequent method invocations by the client results in invocation of the corresponding methods of intercepting plugins in the order specified; and,

an interface that allows a client application to access said implementation.